

CLAIMS

1. In a device that uses a Java programming language and a second programming language, a method of emulating more simultaneously open files in the device than is permitted by an operating system of the device comprising:
- 5 while running a predetermined program in the Java programming language, attempting to access a file stored within the device;
- determining whether the file is already open from prior execution;
- 10 if the file is open, access to the file proceeds;
- if the file is not open, a determination is made by the device of a number of files the device permits to be simultaneously open;
- 15 if the number is not exceeded by opening the file, the file is opened;
- if the number is exceeded by opening the file, the device emulates having more files open than the number allowed by:
- (1) saving a file position for at least one open file, the file position designating where a next byte in the at least one
- 20 open file would be accessed;
- (2) closing the at least one open file; and
- (3) opening the file;
- wherein subsequent accesses to the at least one open file that has been closed are made transparently to the predetermined program
- 25 in the Java programming language by closing at least one currently

open file, retrieving the file position previously saved, and opening and restoring the file position for that file.

2. The method of claim 1 further comprising:

5 saving the file position in storage allocated for usage by Java code.

3. The method of claim 1 further comprising:

implementing the second programming language as C  
programming language.

4. The method of claim 1 further comprising:

providing a table;

storing arbitrarily long file names in a first format;

storing corresponding shortened names in a second format not

exceeding a maximum number of characters; and

translating from the first format to the second format when a file is  
initially opened under control of the Java programming  
language, the translating being implemented transparent to  
any Java programming language application so that the Java  
programming language does not utilize the corresponding  
shortened names.

5. The method of claim 4 further comprising:  
performing the translating only when the operating system does  
not permit file names having a length exceeding a  
predetermined maximum number of characters.

5

6. The method of claim 4 further comprising:  
performing the translating only when the operating system does  
not support first format file names.

- 10 7. The method of claim 4 further comprising:  
placing a unique identifier within predetermined positions of the  
shortened names to identify which file is being retrieved by  
the operating system.

- 15 8. The method of claim 7 further comprising:  
using the unique identifier to bypass the translating of the first  
format to the second format when a predetermined file is  
reopened after being previously closed.

- 20 9. The method of claim 1 further comprising:  
providing a table;  
storing a directory in the table, the directory indicating a Java  
language application suite corresponding to a predetermined  
file; and

using the directory in the table to distinguish files of the same name belonging to different Java language application suites.

- 5    10.    The method of claim 1 further comprising:  
          using storage allocated for use by Java code to contain variables  
          such as file position needed by the second programming  
          language.

- 10   11.    The method of claim 10 further comprising:  
          using the variables to track the current position within an open file  
          and to reopen and reposition the file if the file is temporarily  
          closed.

- 15   12.    The method of claim 10 further comprising:  
          using automatic memory management features of the Java  
          programming language including garbage collection to  
          reclaim storage when the file is no longer in use.

- 20   13.    The method of claim 1 further comprising:  
          implementing the device as a portable, wireless device.

14.    A method of interfacing a file system in a device that uses both a Java programming language and another programming language, comprising:

- 25            selecting a predetermined operating system;

overlying a Java virtual machine for executing programs within the  
operating system in the another programming language;  
overlying the operating system with a file system interface layer;  
overlying libraries to the Java virtual machine and the file system  
5 interface layer, the libraries running in the Java  
programming language;  
using the file system interface to determine if a maximum number  
of open Java files has been exceeded, and if so, storing  
position information of at least one open Java file in a  
10 currently open Java application prior to closing the at least  
one open Java file; and  
subsequently reopening the at least one open Java file by restoring  
the position information transparent to any Java application.

- 15 15. The method of claim 14 further comprising:  
providing a table;  
storing a directory in the table, the directory indicating a Java  
language application suite corresponding to a predetermined  
file; and  
20 using the directory in the table to distinguish files of the same  
name belonging to different Java language application  
suites.
16. The method of claim 15 further comprising:  
25 using storage allocated for use by Java code to store variables  
needed by the another programming language.

17. The method of claim 16 further comprising:

using the variables to track the current position within an open file  
and to reopen and reposition the file if the file is temporarily  
closed.

5

18. The method of claim 14 further comprising:

providing a table;

storing arbitrarily long file names in a first format;

storing corresponding shortened names in a second format not

exceeding a maximum number of characters; and

translating from the first format to the second format when a file is  
initially opened under control of the Java programming  
language, the translating being implemented transparent to  
any Java programming language application so that the Java  
programming language does not utilize the corresponding  
shortened names.

19. The method of claim 18 further comprising:

placing a unique identifier within predetermined positions of the

shortened names to identify which file is being retrieved by  
the operating system.

20. The method of claim 19 further comprising:

using the unique identifier to bypass the translating of the first

format to the second format when a predetermined file is  
reopened after being previously closed.